Declarative Rules for Linked Data Generation at your Fingertips!*

Pieter Heyvaert, Ben De Meester, Anastasia Dimou, and Ruben Verborgh

Ghent University – imec – IDLab,
Department of Electronics and Information Systems, Ghent, Belgium
pheyvaer.heyvaert@ugent.be, ben.demeester@ugent.be,
anastasia.dimou@ugent.be, and ruben.verborgh@ugent.be

Abstract. Linked Data is often generated based on a set of declarative rules using languages such as R2RML and RML. These languages are built with machine-processability in mind. It is thus not always straightforward for users to define or understand rules written in these languages, preventing them from applying the desired annotations to the data sources. In the past, graphical tools were proposed. However, next to users who prefer a graphical approach, there are users who desire to understand and define rules via a text-based approach. For the latter, we introduce an enhancement to their workflow. Instead of requiring users to manually write machine-processable rules, we propose writing human-friendly rules, and generate machine-processable rules based on those human-friendly rules. At the basis is YARRRML: a human-readable text-based representation for declarative generation rules. We propose a novel browser-based integrated development environment (IDE) called *Matey*, showcasing the enhanced workflow. In this work, we describe our demo. Users can experience first hand how to generate triples from data in different formats by using YARRRML's representation of the rules. The actual machine-processable rules remain completely hidden when editing. Matey shows that writing human-friendly rules enhances the workflow for a broader range of users. As a result, more desired annotations will be added to the data sources which leads to more desired Linked Data.

1 Introduction

Linked Data is often generated based on data derived from certain data sources. Initially, custom tools and scripts were used that incorporate directly in their implementation how Linked Data is generated. Updating the semantic annotations resulted in dedicated software development cycles to adjust the implementations. This was circumvented through the use of rules that are defined according to a specific language syntax, such as R2RML [2] or RML [3].

* The described research activities were funded by Ghent University, imec, Flanders Innovation & Entrepreneurship (AIO), the Research Foundation – Flanders (FWO), and the European Union.

These languages define *declaratively* how Linked Data is generated from corresponding data sources using annotations provided through vocabulary terms. Rules are detached from the implementation that executes them, thus, the implementation does not need to be updated when the rules are updated. As such Linked Data generation languages are built foremost with machine-processability in mind, it is not always straightforward for users to define or understand rules written in these languages. This prevents the users from specifying the desired annotations for the data sources. In the past, graphical editors were proposed, e.g., the RMLEditor [4] and Map-On [7], to enhance the workflow of defining rules. However, next to users who prefer a graphical approach, there are users who desire a text-based approach. For the latter, we introduce an enhancement to their workflow. At the basis of this enhancement is YARRRML[1], a human-readable text-based representation for declarative generation rules.

To investigate a human-readable text-based representation in the workflow, we propose a novel browser-based integrated development environment (IDE) called *Matey*[2]. Even though other IDEs and text editors can be used to work with YARRRML, Matey showcases the enhanced workflow of the rules generation process, such as the samples of the data sources, the generation of the machine-processable generation rules, and the corresponding Linked Data. Through the use of YARRRML, the underlying languages' complexity and verbosity are hidden.

In this work, we describe our demo, during which participants can have a hands-on experience with Matey to define machine-processable rules from data in different formats by using YARRRML's representation of the rules. With Matey we show that a broader range of users can enhance their workflow for defining rules via writing human-friendly rules. As a result, more desired annotations will be added to the data sources which leads to more desired Linked Data. In Section 2, we briefly summarize YARRRML, and in Section 3, we discuss and demonstrate Matey. Matey is available at `https://w3id.org/yarrrml/matey/` and a screencast is available at `https://w3id.org/yarrrml/matey/screencast`.

2 Human-readable text-based representation

YARRRML is a human readable text-based representation for declarative Linked Data generation rules. It is expressed in YAML [1], a widely used data serialization language designed to be human-friendly. It is already specified how YARRRML can be used to represent R2RML and RML rules. Through the example in Listing 1, we summarize YARRRML's basic concepts[3].

All rules that state how subjects, predicates, and objects are generated are found under the `mappings` key, which is attached to the root of the YARRRML document (Listing 1, line 4). Per set of rules that state how an entity is generated together with its corresponding attributes, a user-chosen key is added to the

[1] `https://w3id.org/yarrrml/`
[2] From *mate* + *-y* (pronounced *M-eighty*), i.e., a fellow pirate.
[3] The specification is available at `https://w3id.org/yarrrml/spec/`

```
1  prefixes:
2    ex: "http://example.com/"
3
4  mappings:
5    person:
6      sources:
7        - ['data.json~jsonpath', '$.persons[*]']
8      s: ex:$(firstname)
9      po:
10       - [a, foaf:Person]
11       - [foaf:givenName, $(firstname)]
```

Listing 1: example YARRRML rules that annotate the data in Listing 2

```
1  {
2      "persons": [
3          {"firstname": "John",  "lastname": "Doe"     },
4          {"firstname": "Jane",  "lastname": "Smith"   },
5          {"firstname": "Sarah", "lastname": "Bladinck"}
6      ]
7  }
```

Listing 2: example JSON file (data.json) detailing people's first and last name

mappings key. In Listing 1, you can find such a set of rules for the generation of Linked Data from the JSON file in Listing 2. The file contains metadata information about people, including their first and last name. The user-chosen key person has as value all the rules related to the entity that represents a person. That key can be reused in other rules when there is a relationship between the different entities. The key sources has as value all the data sources that are used to generate the person entities, which includes the name of the file and an optional iterator. The latter determines the records that represent the different entities.

The key s has as value the rules that state how subject-IRIs are generated for the different entities (Listing 1, line 8). In this example, each IRI is constructed by appending the first name of every person to http://example.com/. The key po has as value the rules that state how combinations of predicates and objects are generated (Listing 1, line 9). For example, the rule at Listing 1, line 10 states that the class of every person is foaf:Person. The rule at line 11 states that for every person the value in the JSON attribute firstname is related to a person via the predicate foaf:givenName[4].

3 Matey

YARRRML's Matey is a browser-based IDE[5] for viewing and defining Linked Data generation rules in a YARRRML representation, while the corresponding

[4] The prefix foaf is short for `http://xmlns.com/foaf/0.1/`, as YARRRML by default includes the predefined prefixes of RDFa (see `https://www.w3.org/2011/rdfa-context/rdfa-1.1`).

[5] `https://w3id.org/yarrrml/matey/`

RML rules can be exported. Additionally, the rules can be executed in Matey on a sample of the data, which allows users to inspect the generated Linked Data. Through the use of a YARRRML representation the underlying language's complexity and verbosity are hidden. Although other IDEs and text editors can be used to work with YARRRML, Matey pays special attention to the specific aspects of the rules generation process, such as the samples of the data sources, the generation of the machine-processable generation rules, and the corresponding Linked Data.

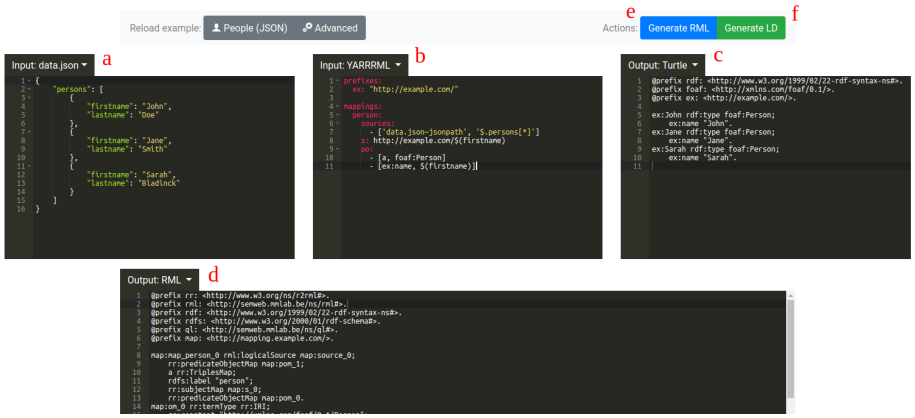

Fig. 1: GUI of Matey

The Graphical User Interface (GUI) of Matey is visible at Figure 1. It fulfills the seven requirements for GUIs for the creation of Linked Data generation rules we introduced in previous work [5]: is independent of the underlying language (R1) and rule execution (R2); supports multiple data sources (R3), heterogeneous data formats (R4), and multiple ontologies (R5); and enables multiple alternative modeling approaches (R6) and non-linear workflows (R7).

The GUI consists of two rows: the top row contains three panels (the editing area), and the bottom row contains a single panel (the results). Matey's top row follows the RMLEditor's layout [4]. The **top left panel** is an editing area showing a sample of the data sources from which Linked Data is generated (see Figure 1, a). Multiple data sources can be added through the use of a drop down menu (R3) with data in different formats, such as CSV, JSON, and XML (R4).

The **top middle panel** is an editing area showing the YARRRML representation of the rules (see Figure 1, b). The representation is independent of RML, the underlying mapping language (R1). It automatically generates the RML rules based on the YARRRML representation, and no restrictions are enforced by the GUI on which and how many ontologies are used (R5).

In the **top right panel**, the resulting Linked Data is shown (see Figure 1, c).

In the **bottom panel**, the RML rules corresponding with the YARRRML representation are shown (see Figure 1, d). They can be exported and reused by existing tools supporting such rules (R2). Thanks to the layout of these panels, users can follow different rules creation approaches, such as the data-driven and schema-driven approach [6] (R6). Furthermore, users can inspect the panels, and optionally update their content, independently at any time (R7).

During an example workflow, users add their data in the top left panel and define the rules in the middle panel. Next, either they generate the corresponding RML rules by clicking "Generate RML" (see Figure 1, e) or they generate Linked Data by clicking "Generate LD" (see Figure 1, f). During the former, the rules appear in the bottom panel. During the latter, the generated Linked Data appears in the top right panel. In case the Linked Data is not as desired, users update the rules in the middle panel. Once users are satisfied with the rules, they export them as YARRRML and RML rules via their respective panels.

During the demo participants will be able to have a hands-on experience with Matey. They will be able to define their own rules on their own data. Matey is publicly available at `https://w3id.org/yarrrml/matey/` and a screencast is available at `https://w3id.org/yarrrml/matey/screencast`. The demo showcases – next to manually defining machine-processable rules and using a graphical tool – a textual human-friendly alternative to understanding and defining rules. The complexity of the machine-processable rules are hidden and remain interoperable with existing tools that use Linked Data generation rules.

References

1. O. Ben-Kiki, C. Evans, and B. Ingerson. YAML Ain't Markup Language (YAML) Version 1.2. Technical report, 2009.
2. S. Das, S. Sundara, and R. Cyganiak. R2RML: RDB to RDF Mapping Language. Working group recommendation, World Wide Web Consortium (W3C), Sept. 2012.
3. A. Dimou, M. Vander Sande, P. Colpaert, R. Verborgh, E. Mannens, and R. Van de Walle. RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data. In *Proceedings of the 7th Workshop on Linked Data on the Web*, volume 1184 of *CEUR Workshop Proceedings*, 2014.
4. P. Heyvaert, A. Dimou, B. De Meester, T. Seymoens, A.-L. Herregodts, R. Verborgh, D. Schuurman, and E. Mannens. Specification and implementation of mapping rule visualization and editing: MapVOWL and the RMLEditor. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2018.
5. P. Heyvaert, A. Dimou, R. Verborgh, E. Mannens, and R. Van de Walle. Towards a Uniform User Interface for Editing Mapping Definitions. In *Proceedings of the 4th International Workshop on Intelligent Exploration of Semantic Data (IESD 2015)*. CEUR-WS.org, 2015.
6. P. Heyvaert, A. Dimou, R. Verborgh, E. Mannens, and R. Van de Walle. Towards Approaches for Generating RDF Mapping Definitions. In *Proceedings of the ISWC 2015 Posters & Demonstrations Track*. CEUR Workshop Proceedings, 2015.
7. Á. Sicilia, G. Nemirovski, and A. Nolle. Map-On: A web-based editor for visual ontology mapping. *Semantic Web*, 8(6):969 – 980, 2017.